Remarks:

This response is directed to the Office Action mailed September 7, 2004. Claims 1, 25, 34, 35, 40, and 47 have been amended. Applicant respectfully requests reconsideration and withdrawal of the rejections in view of above amendment and the following remarks.

Claim Objections:

The examiner asserts in the Office Action that a "computer process" has a particular standard meaning in the art: "i.e., a task that is scheduled for execution by an operating system." While Applicant recognizes that this may be one recognized definition, the examiner has provided no basis for his assertion of such standardization. It is respectfully submitted that Applicant's terminology of "computer implemented method" includes, but is not limited to, a "computer process" as defined by the examiner.

Rejection under 35 USC 102(b): Anticipation by Jagannathan:

Claims 1-6, 15-26, 31-32, and 34-51 stand rejected under 35 USC 102(b) as anticipated by Jagannathan et al (USPN 5,692,193). The examiner asserts that Jagannathan teaches the invention as claimed, summarizes that claim, and then summarizes the claim limitations, referring to portions of the Jagannathan et al specification that he believes provides the anticipatory disclosures for the limitations in each independent and dependent claim.

It is respectfully submitted that there is a fundamental difference between the claimed "thread-specific heaps" in Applicant's application and the "local heaps" disclosed in the Jagannathan patent. They are not the same and do not operate in the same way. The objects placed in Applicant's thread-specific heaps are **guaranteed** to be local to a thread whereas the objects placed in Jagannathan's local heaps are **presumed** to be local to a thread. There is simply no suggestion of such "front end" verification of local thread heap objects in Jagannathan et al.

In Jagannathan's system, if the presumption turns out to be wrong and therefore a reference to an object in a local heap somehow escapes, the object (and other objects transitively reachable from the object) must be copied into a global heap.

Jagannathan states, at column 21, lines 44-52:

This is because any object that is referenced from a shared object is also a shared object and, therefore must reside in a shared heap. This constraint on shared heaps is

enforced by ensuring that references stored in shared heaps refer to objects that are (a) either in a shared heap, or (b) <u>allocated in a local heap and garbage collected into a shared one.</u> (Emphasis added) That is, the graphs of objects reachable from the referenced object must be copied into or located in the shared heap. The overheads of this memory model depend on how frequently references to objects allocated on local heaps escape.

This operational system is based on a presumption. That is, that in Jagannathan's system, the objects placed in local heaps are PRESUMED to be local to a thread. The runtime checks for violations of the assumption of thread locality of objects have a cost; most likely the check is incorporated into a write barrier, and the garbage collector must be designed to work with the on-demand copying out of objects from a local heap into a global heap.

In contrast, in Applicant's claimed system, the program analysis performs a proof that the objects in the thread-specific heaps are never accessed by threads other than the one owning the thread-specific heap in which each object resides. Applicant's specification, pages 7, line 14 to page 8, line 18 states:

> Thread-specific data is distinguished from shared data in that **thread-specific data is determined to be unreachable from outside of a given program thread.** An exemplary method of determining whether program data is deemed potentially reachable from outside of a given program thread is called "thread escape analysis", although other analysis methods are also contemplated within the scope of the present invention. **Program data that is not determined to be "thread-specific data" is deemed "shared data".** As such, shared data includes program data that is deemed potentially reachable from multiple program threads. Furthermore, a thread-specific heap may exist for each program thread of a target program, and program data specific to each program thread are allocated appropriately to the associated thread-specific heap.
>
> During runtime, thread-specific data is allocated to thread-specific heaps, and shared data is allocated to one or more shared heaps. At some appropriate time during execution, a garbage collector module reclaims memory for unneeded program data from one or more of the target program's heaps. **Because no thread-specific data within a thread-specific heap is determined to be reachable from outside the associated program thread, the thread-specific heap can be collected without impacting (e.g., suspending or synchronizing with) the execution of other program threads in the target program.** In addition, the impact of garbage collection of the shared heap on program threads can be minimized because the program threads can substantially continue their execution so long as they do not access program data in the shared heap.

It should be understood that exemplary program analysis that determines "reachability" at compile time is based on a conservative estimate of what may occur at runtime. Therefore, in an embodiment of the present invention, the **program analysis at compile time proves that an object is "thread-specific" by proving that the object will never be referenced by multiple threads. However, an object may be deemed potentially reachable by multiple threads (i.e., "shared"), if it cannot be proven that the object will never be referenced by multiple threads (i.e., even though it has not been proven that the object will indeed be referenced by multiple threads during execution). In contrast, during runtime, reachability can be specifically determined, thereby making the conservative approximation unnecessary during runtime.** (emphasis added)

, and page 11, lines 17-19:

An object that is proven to be reachable by only a single program thread is referred to as a "thread-specific object". In contrast, if the object is deemed potentially reachable by more than one program thread, the object is referred to as a "shared object".

Therefore, in Applicant's invention as claimed, there is **no need** for any checks of violations of an assumption of thread locality of objects. In addition, the garbage collector algorithms for the thread-specific heaps do not need to allow for on-demand copying out of objects from a thread-specific heap into a global heap as in Jagannathan. This differs substantially from Jagannathan.

The examiner's position is that Jagannathan suggests using analysis to guide where to allocate objects (Jagannathan, column 21, lines 55-57). This position is correct. The important thing to note, however, is that the program analysis is used for **opposite** purposes and leads to opposite results. Jagannathan suggests using program analysis **during runtime** to determine which objects **are likely** to escape and should therefore initially be allocated in a shared heap rather than initially being allocated in a local heap and then subsequently copied out into a shared heap. In Applicant's system, program analysis is used **at compilation time** to **prove** that certain objects are thread-specific such that runtime analysis is NOT called for, as "escape" is not possible. In contrast, Jagannathan makes a presumption and does not prove thread specificity.

Applicant has amended claims 1, 25, 34, 35, 40 and 47 to specifically recite that the thread-specific data is **proven** thread specific data.

Applicant's Claim 1 now states:

A computer program product encoding a computer program for executing on a computer system a computer implemented method for managing allocation of program data in a target program between one or more thread-specific heaps and at least one shared heap, the program data including thread-specific data and shared data, the computer implemented method comprising:

analyzing the target program during code compilation to distinguish between **proven** thread-specific data of a first program thread and the shared data;

configuring the target program to allocate the thread-specific data of the first program thread to a first thread-specific heap, responsive to the analyzing operation; and

configuring the target program to allocate the shared data to the shared heap, responsive to the analyzing operation.

Independent claims 1, 25, 34, 35, 40 and 47 each recite "**proven** thread-specific data" as defined precisely in Applicant's specification. Therefore it is respectfully submitted that Jagannathan et al does not disclose or teach Applicant's CLAIMED invention as defined in these independent claims. Accordingly the rejection under 35 USC 102(b) of claims 1-6, 15-26, 31, 32, and 34-51 should now be withdrawn. It is respectfully submitted that the amended language is inherently present in the originally presented claims in view of the specific definition of the term "thread-specific data" in Applicant's specification, and therefore the amended language was not earlier presented. The amended language is specifically provided to more clearly emphasize the nature of the claimed thread-specific data, such that a further search or consideration is clearly not warranted.

Rejection under 35 USC 103(a)

Dependent claims 7-14, 27-30, and 33 stand rejected as obvious over Jagannathan in view of Benayon et al. The examiner agrees that Benayon does not specifically address identifying thread specific objects and allocation to thread specific heaps during compilation, but asserts that this step is taught by Jagannathan et al. As discussed above, the analysis which proves that an object is thread specific is not performed in Jagannathan et al. A presumption is merely made. Since the rejected claims depend from Applicant's amended independent claims 1, 25, 34, 35, and 40, and Benayon et al does not disclose **proven thread-specific data** as claimed, the rejection of the dependent claims 7-14, 27-30, 33 should be withdrawn. These claims are clearly believed to be patentable for the reasons set forth above.

   In view of the above amendments and remarks, Applicant respectfully requests a Notice

of Allowance. If the Examiner believes a telephone conference would advance the prosecution

of this Application, the Examiner is invited to telephone the undersigned at the below-listed

telephone number.

                                   Respectfully submitted,
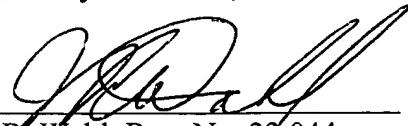
_____10/20/04_____   _____
Date   John R. Wahl, Reg. No. 33,044
                                   Merchant & Gould P.C.
                                   P.O. Box 2903
27488   Minneapolis, MN 55402-0903
PATENT TRADEMARK OFFICE   (303) 357-1644
                                   (303) 357-1641 (fax)